Robotics Practicals

Topic 1: Application of Bayes filters to mobile robot localization

Jan FROGG, Sascha FREY, Stanislas FURRER

March 11, 2019

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

1 Introduction

It is important for an autonomous robot to be able to localize itself in any environment. It must be able to detect its own position in a given map in order to then be able to interact with it and to perform a set of tasks without hitting an obstacle for instance. The goal of this practical work is to understand the application of a **Bayesian filter** to solve the localization problem in robotics. To do so, we will be looking at one type of Bayesian filtering, called the **Monte Carlo Localization**, by implementing it using C++ on a real Thymio robot after testing it in a simulated environment. In this report we will first describe the theoretical aspects of the experiment by explaining the method used and answering theoretical questions before diving into the results and their analysis.

2 Theory

2.1 Method

The Monte Carlo Localization method is a type of Bayesian filter, which means it relies on a probabilistic model to tell where the robot is in the map using a set of samples called **particles**. Each particle contains a guess on the robot's state (x_t, y_t coordinates and θ_t orientation in our case). This is done in two steps: the ***Time update*** (prediction) step which computes the probability of a state based on odometry information and ***Measurement update***(correction) which updates the probability of a state based on sensor information. These probabilities are represented by their occurrence in the set of particles.

2.2 Theoretical questions

2.2.1 Question 1

In some cases (for instance when the number of particles is low) it can happen that after an unlucky re-sampling there are no particles close to the correct state, it is called the **particle deprivation problem**. To correct for this, we include some randomness in the model by taking a fraction of the total number of particles (10% in this case) and shuffling them constantly so that there always a way to re-sample them in order to land on the correct state, even if the previous re-sampling gave a wrong state estimation. This is equivalent to assuming that, at any point in time, the robot has some small probability of being "kidnapped" to a random position in the map, thus causing a fraction of random states in the motion model. Moreover, we add some white noise around the calculated mean value when updating a particle in both the time and measurement update steps. This again allows for more flexibility and robustness by preventing the algorithm to pick one wrong state and accumulating erroneous behaviour.

2.2.2 Question 2

The motion model $p(x_t|u_t, x_{t-1})$ defines robot motion in an empty space. However when we know the map we can take it into account and consider the probability $p(x_t|u_t, x_{t-1}, map)$. In general we say that $p(x_t|u_t, x_{t-1})$ is not equal to $p(x_t|u_t, x_{t-1}, map)$. The reason for this is that the motion model is based on the odometry information only, which carries some uncertainties such as wheel errors on size and velocity, initial position of the robot and ground errors i.e. the robot might be placed on a slippery surface, which results in an wrong position or speed measurement. The map provides additional information based on the sensors which allows to compensate for these errors coming from the motion model. One example of the need for such compensation is when a Thymio-like robot founds itself in a map after performing a straight-line vertical movement (observer is above the map looking down) and a little angle is induced from one wheel having a slightly higher speed than the other one. While the information from both wheels state that the robot has had a straight movement, information from the map is important here to tell the robot it has in fact turned a bit by the aforementioned angle, so it can correct for this and adapt the localization process.

3 Experimental results

3.1 Data logs

The experiment consisted of running the code on the real Thymio robot (after simulation) for 10 trials - 5 trials using 10'000 particles and 5 trials using 50'000 particles. In order to let the robot perform the localization process, we drove it around the map using the same trajectory for every single trial as a way to have a good basis for comparing the performance between each trial. The trajectory we chose was sort of a "banana" shaped one, starting at the top left corner of the map and finishing in the top right corner, and then going back to end the trial. This trajectory was chosen to maximize the success rate of localization, the algorithm encountered problems when on a purely white space.

While performing these tasks, continuous recordings of position and angle were received through a data log. Three different logs were recorded:

- **Reference**: log of the robot's states at certain time steps for the code to which we will compare our algorithm.
- **Estimation**: log of the robot's states at certain time steps for our code.
- **Tracking**: visual tracking of the robot from a camera to establish a ground truth which our algorithm must follow in a reliable manner.

3.1.1 Data Preparation

The data was processed using a Matlab script. The largest difficulties in preparing the data were in the synchronization of the time steps and the removal of outlier behaviour. The first 20 seconds of collected data were removed as these were used to calibrate the camera used to get the "true" tracking position. Furthermore, points that had a very large error with respect to the true state were removed as these were considered to be outliers.

The tracking log is sampled at a rate approximately 10 times larger than the other two logs, furthermore samples are not aligned in terms of time. The missing points were therefore completed by simply using linear interpolation.

3.2 Performance

We first analyze the data from all 3 logs when performing the experiment using 10'000 particles (first 5 trials). We start by looking at the trajectory which is shown in Figure 1. As we can see, our estimation seems to be following the reference and tracking in a reliable manner overall, although there are a few points which do not follow the true trajectory, especially at the beginning and at the end of the path traveled by the robot. The 3^{rd} trial shows a significantly erroneous behaviour of the algorithm, showing that both our implementation and the reference implementation can be improved in terms of robustness. However, looking at the remaining trajectories the reference algorithm often exhibits superior performance.

Secondly, we look at the average (over all 5 trials) error on the position (Euclidean Distance) and angle of the robot on the map between our estimation and the tracking as well as between the reference and the tracking. These errors are plotted as a function of distance as visible in Figure 2. The travelled distance was calculated by finding the cumulative sum of the distance travelled between each step. Both the reference and estimation algorithm follow a similar pattern of error with respect to the tracking signal for position and angle. They both start at a certain value and decrease at the same rate towards a near-zero value as the localization algorithm does its job. The estimation-to-tracking position error, however, shows two small spikes around a distance of 0.5m and another spike around a distance of 1.7m. The reference-to-tracking position error also has one slight peak around a distance of 0.8m. When looking at the error on the orientation of the robot, Figure 2 indicates that again both signals decrease at the same rate to nearly zero and both of them show some spikes and single peaks (outliers) at various distances. The likely explanation for these spikes is the white space which provides insufficient information to the algorithm to effectively compute the measurement update. Effectively the algorithm assigns equal probability to multiple locations, all having a predominantly white surface. The results are the location of the highest probability location which changes easily due to the close probabilities. This results in the robot position jumping all over the place. Overall, this shows that while our algorithm for the estimation might need some optimization, it is still really close to that of the reference. Furthermore, the error plots show that the error is mainly kept under 0.1m, with large portions under 0.05m which indicates fairly good performance.

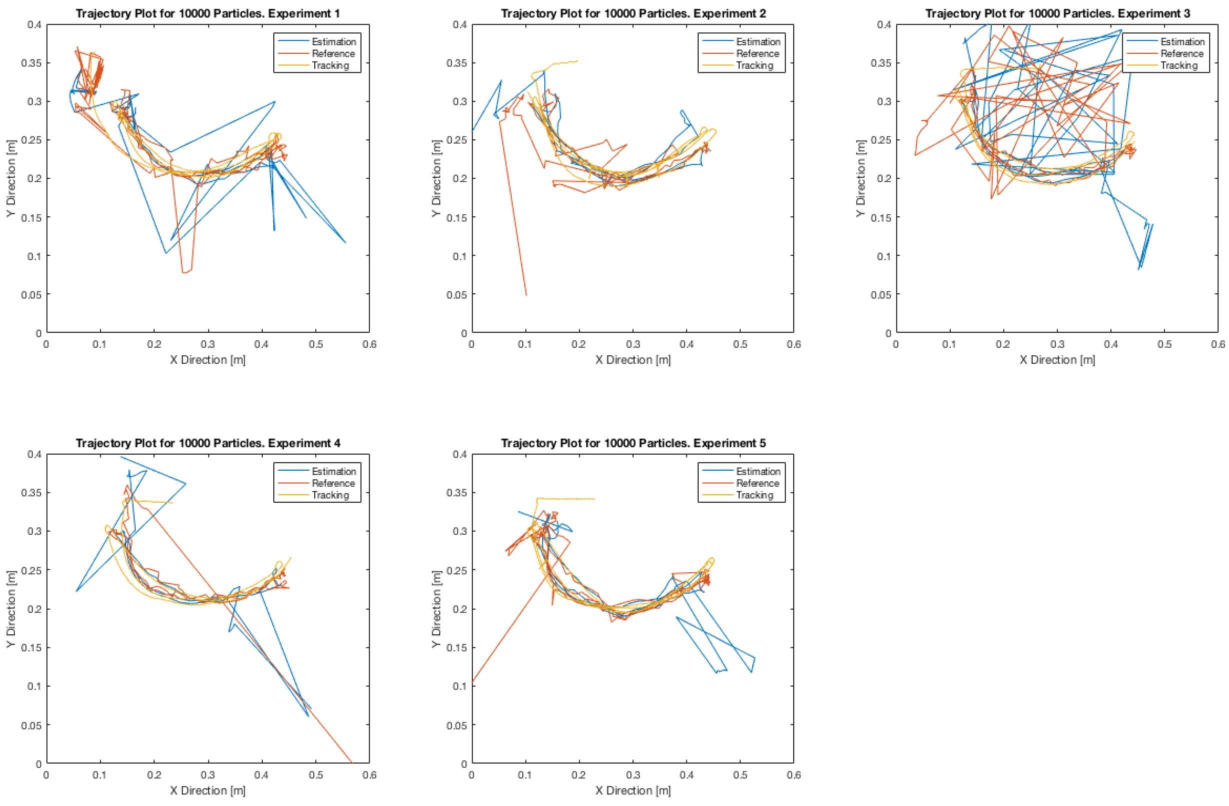


Figure 1: Trajectory plots on the map for the experiment using 10'000 particles for all 3 logs, ordered by experiment number from 1 to 5.

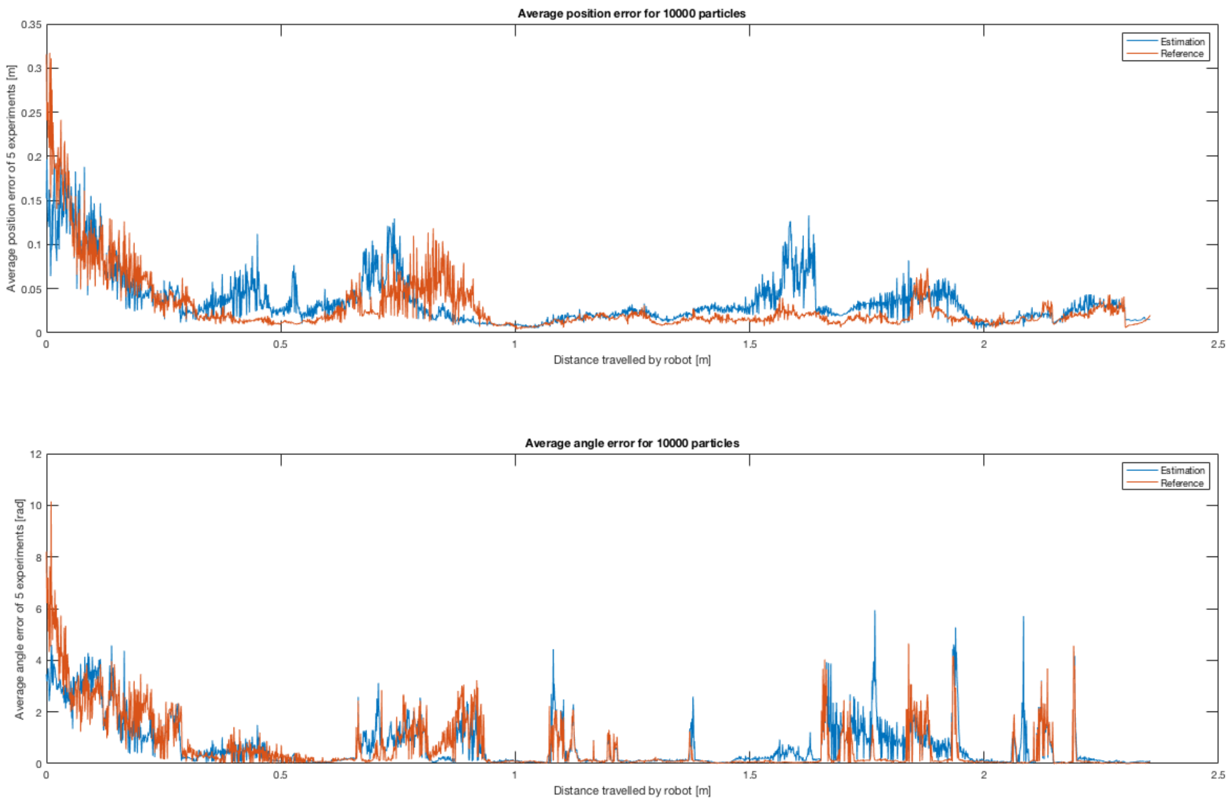


Figure 2: Position and angle error plots for the experiment as a function of the distance ran by the robot using 10'000 particles, ordered by experiment number from 1 to 5.

We now look at the performance of the algorithm using 50'000 particles. An increased amount of estimated states should in theory provide more accuracy to the model, although it will also increase the computational cost of the algorithm. By looking at Figure 3, which represents the same plots as in Figure 1 but now with 50'000 particles, we see that whilst the 3^{rd} trial shows better behaviour of the model, we still have major errors at the starting and ending points of the trajectory, and even along the trajectory, which indicates overall that the algorithm gives at best the same result as in the case of having 10'000 particles. The results are very similar to the 10,000 particle case implying that when the algorithm receives sufficient information it works well with either number, but without sufficient differentiation in the input, increasing the number of particles is not beneficial.

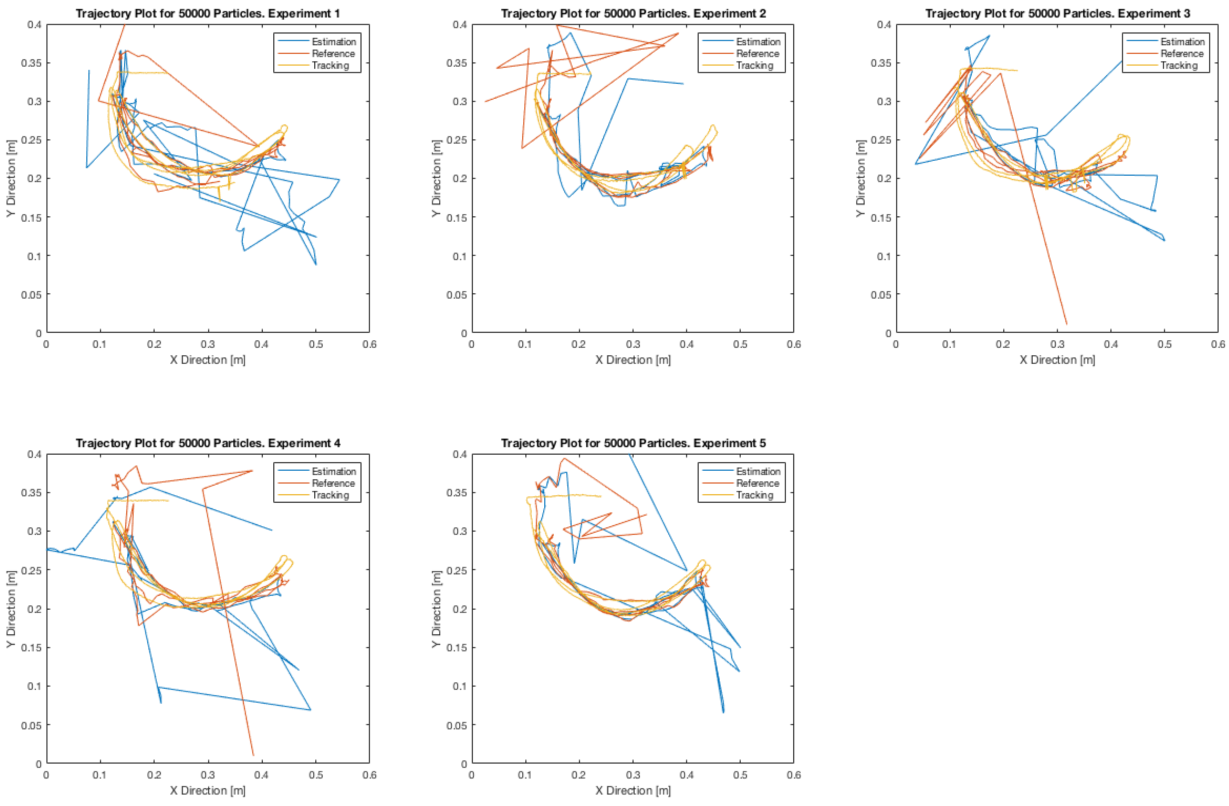
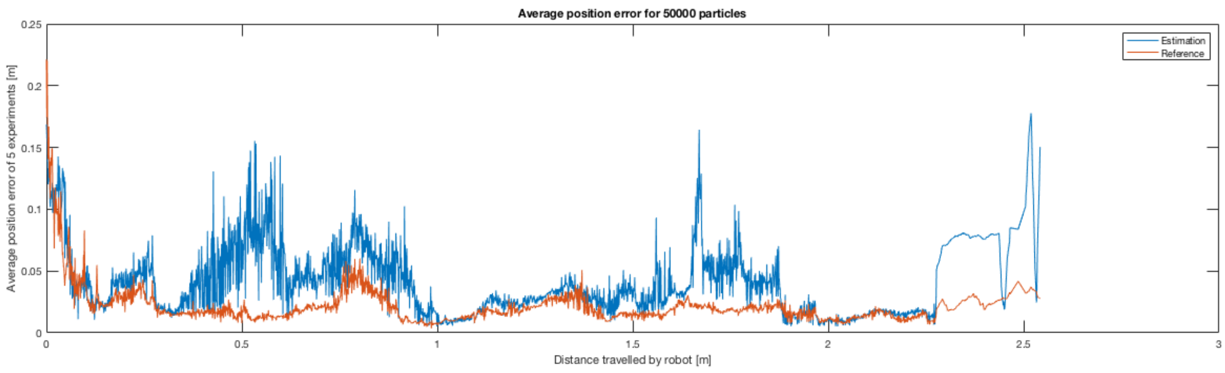


Figure 3: Trajectory plots on the map for the experiment using 50’000 particles for all 3 logs, ordered by experiment number from 1 to 5.

Regarding the reference and estimation -to-tracking errors, as in the case with 10’000 particles we get an error which decreases in the same way for the estimation to that of the reference both for position and orientation, which is satisfying. However, there are similar peaks of error that are even somewhat larger than for the 10’000 particle case. The error orientation seems to have a similar trend both for reference and estimation, where both signals show some peaks at several distances.

In light of these results, we may conclude that either our model is much more sensitive to small variations in position or orientation when we feed it a larger amount of particles.



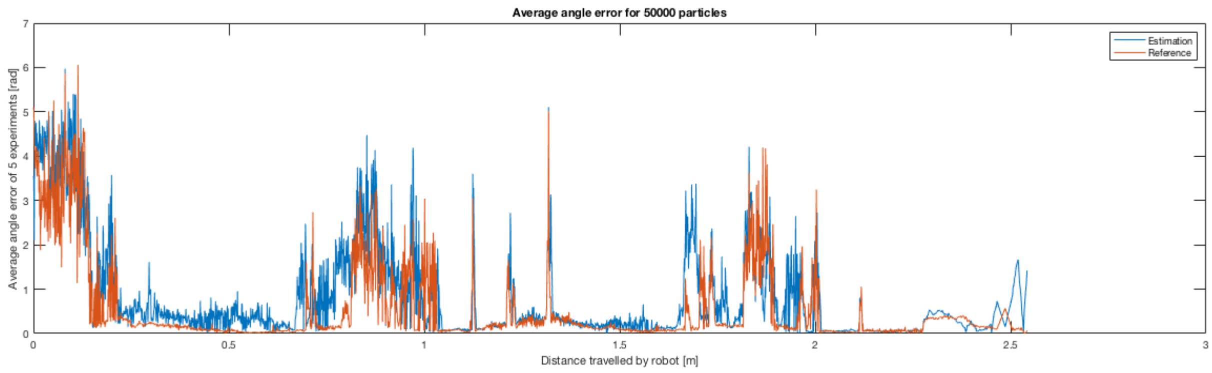


Figure 4: Position and angle error plots for the experiment as a function of the distance ran by the robot using 50'000 particles, ordered by experiment number from 1 to 5.

3.3 Localization results

To further analyze the performance of each implementation of the Monte Carlo Localization algorithm we calculate the mean distance that the robot has to travel before being able to estimate it's location accurately in a stable manner. This was done by finding the first occurrence of 300 data points (out of approximately 2500) that have an error below a certain threshold. The decision to use this consecutive decision rather than a permanent condition arose from the above-mentioned short deviations of the estimated position with respect to the true position. The collected distances are represented in Figure 5.

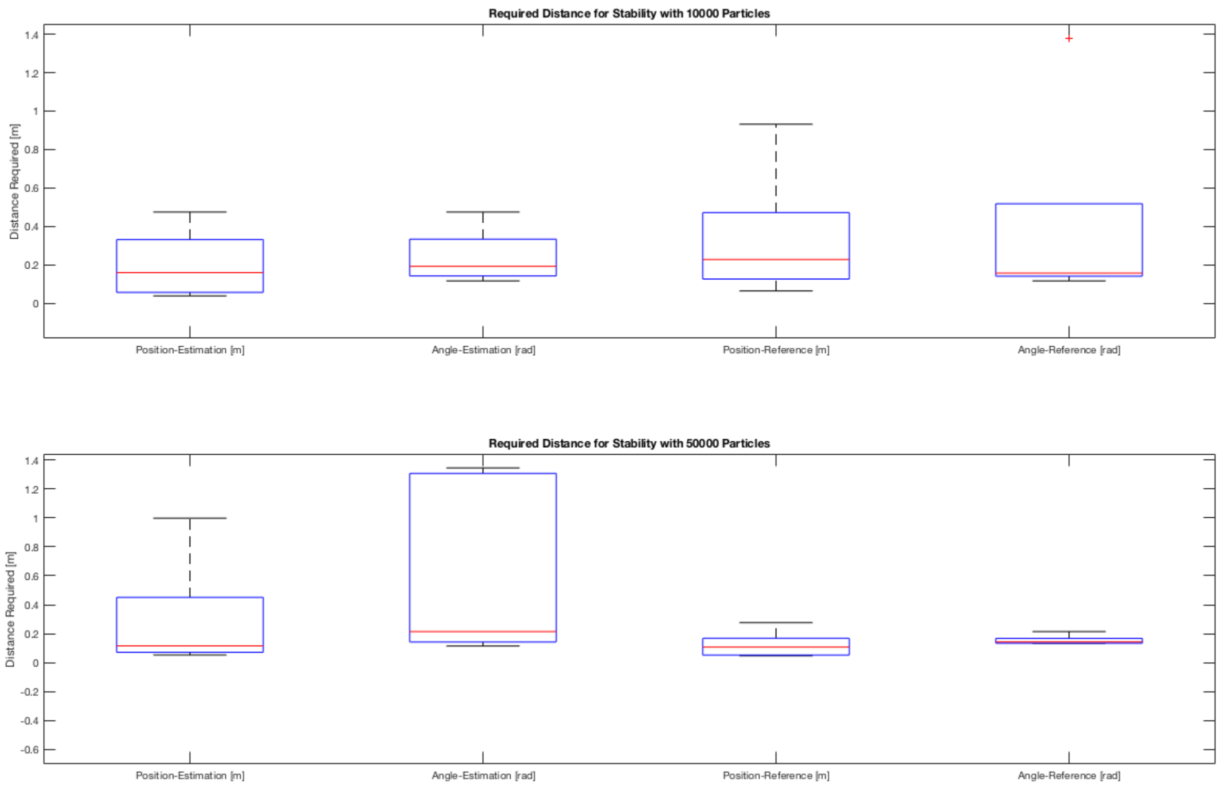


Figure 5: Box plot for the distances covered by the robot until the localization process is stable.

We firstly see a confirmation that the Estimation and Reference algorithms are very similar for 10'000 particles, having similar means and variances. The only significant difference is that there is an outlier point for the reference algorithm, likely from experiment 3.

What is interesting is that the mean required distance remains identical for the two implementations for 50'000 particles, but the estimation implementation exhibits a much larger variance. This likely also explains the deviation seen in Figure 4. For some reason our implementation of the algorithm induces more variability with more particles. At the time of writing it is unclear why this is but we assume it is due to a difference in chosen parameters for the added randomness to increase robustness.

3.4 Algorithm flaws and possible optimization

Before mentioning any optimization step, it is important to point out that some parameters of the code may be tuned in order to observe various behaviours of the model. Such parameters include the fraction of particles which are picked at random, the type of noise added to the re sampling of particles and others. While this tuning offers diversity in the results, the algorithm is most definitely subject to enhancement and optimization overall.

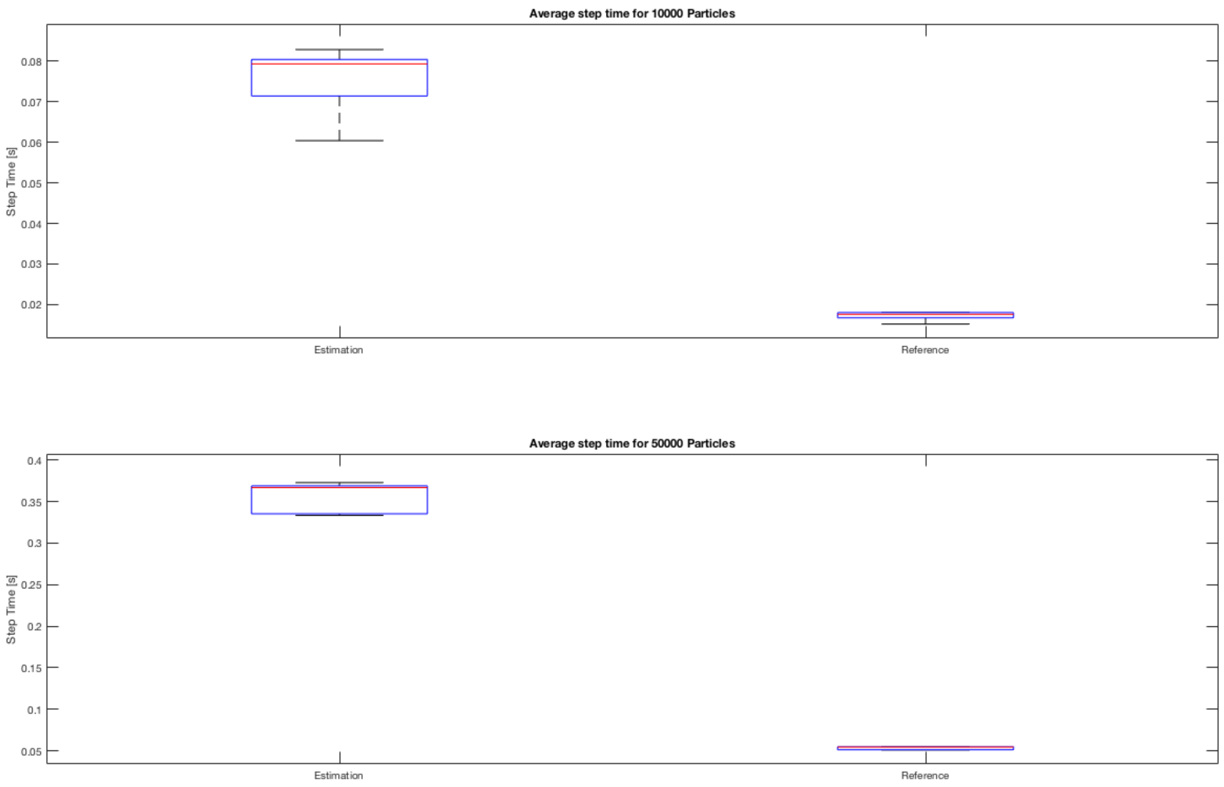


Figure 6: Box plot representing the average computation time (step time) for the estimation and reference and for each situation i.e. using 10'000 and 50'000 particles.

3.5 Conclusion and Suggestions

3.5.1 Suggestions

The test of the algorithm take place on one of the two arenas available. Unfortunately one of them presents many white areas. The Thymio robot has the tendency to get lost when there are too many white points or when they are too big. It may be advisable to adapt the gray scaling of the map with too much white area, or to optimize the code in order to make it more efficient regarding light grayscale values. This would obviously contribute to the robustness of the model.

During the practical it was only asked to evaluate the robustness of the algorithm on a known path consisting of a round trip. It would be interesting to add an experimental part aimed at testing the robustness of the algorithm on a random path.

It would be of great value to add one more session to program the Markov localization algorithm and to compare the end results with the Monte Carlo method. By the way, the programming of the Markov localization is similar and easier than the Monte Carlo one.

3.5.2 Conclusion

Overall we are satisfied with our results. In the case of 10,000 particles, our code has shown similar or even slightly better performance than that of the reference. However, when using 50,000 particles, the variance of our results (the estimation signal) was much greater than that of the reference. In general, our implementation of the algorithm still allowed the Thymio to get a precise idea of its position in a relatively reliable way. This practical was a good opportunity to explore theoretical concepts and to implement them. As usual for a practical work, it showed that while understanding the theory is somewhat simple, it is a completely different story when testing on a real robot, as one has to adjust to various parameters and has to deal with uncertainty and robustness of the model.